Evaluation of Data Management Systems for Geospatial Big Data

Pouria Amirian[1], Anahid Basiri[2], and Adam Winstanley[1]

[1] Department of Computer Science, National University of Ireland Maynooth, Ireland
[2] Nottingham Geospatial Institute, The University of Nottingham, UK
amirian@cs.nuim.ie, anahid.basiri@nottingham.ac.uk
adam.winstanley@nuim.ie

Abstract. Big Data encompasses collection, management, processing and analysis of the huge amount of data that varies in types and changes with high frequency. Often data component of Big Data has a positional component as an important part of it in various forms, such as postal address, Internet Protocol (IP) address and geographical location. If the positional components in Big Data extensively used in storage, retrieval, analysis, processing, visualization and knowledge discovery (geospatial Big Data) the Big Data systems need certain type of techniques and algorithms for management, analytics and sharing.

This paper describes the concept of geospatial Big Data management with focus on using typical and modern database management systems. Then the typical and modern types of databases for management of geospatial Big Data are evaluated based on model for storage, query languages, handling connected data, distribution models and schema evolution. As the results of the evaluations and benchmarks of this paper illustrate there is no single solution for efficient management of geospatial Big Data and in order to utilize unique characteristics of geospatial Big Data (such as topological, directional and distance relationship) a polyglot geospatial data persistence system is needed.

Keywords: geospatial Big Data, graph database, XML document database, column-family database, spatial database, geospatial Big Data Management, polyglot geospatial data persistence.

1 High Level Introduction to Geospatial Big Data

Often data component in Big Data has a geospatial component as an important part of it in various forms, such as postal address, Internet Protocol (IP) address and geographical location (geospatial Big Data). As it mentioned in many research papers, management and analysis of geospatial data is complex and requires specific storage, processing, analysis and publication mechanisms [1, 2, 3, 4, 5 and 6]. In fact management and analysis of geospatial data have been always revealed the limitations of information systems and computational frameworks. In a nutshell, unique characteristics of geospatial data such as high volume, various type of relationships between geospatial objects (e.g. distance, directional and topological relationships), need for long transactions, computationally intensive algorithms of processing and inclusion of

B. Murgante et al. (Eds.): ICCSA 2014, Part V, LNCS 8583, pp. 678–690, 2014.
© Springer International Publishing Switzerland 2014

time component, makes the management and analysis of geospatial Big Data even more complicated. Some researchers agreed that geospatial data may represent the biggest Big Data challenge of all [7]. If the positional components in Big Data extensively used in storage, retrieval, analysis, processing, visualization and knowledge discovery (geospatial Big Data) the Big Data systems need certain type of technologies, techniques and algorithms for management, analytics and sharing. [8]. Using geospatial Big Data provides unprecedented opportunities for providing improved, more adaptive, more intelligent and cost-effective services in government, private and science and research sectors [9]. In summary management of geospatial data has several challenges in the storage, processing, analysis, visualization and publication areas. This paper focuses on management of geospatial Big Data for standard online sharing and publication.

2 Standard Publication of Geospatial Big Data Using Web Services

Publication of geospatial Big Data in standard manner provides opportunities for executing distributed and collaborative data preparation, data mining and knowledge discovery tasks. Also the ever-increasing access to geospatial data on the Web results in enhanced system efficiency through cost and time reduction in data collection, data preparation and information retrieval. Moreover, such access helps decision-makers to manage their assets better, enables faster responses for time-sensitive decisions, and improves the communication process across diverse agencies. In this regard, geospatial data should be shared and accessed using standard services which are openly published over the Web [10]. In this context, there are generally two approaches for publishing geospatial data in standard manner. The first approach is to use the specifications published and managed by Open Geospatial Consortium (OGC). The mentioned specifications (geospatial services) consist of defined set of request/responses to access geospatial resources [11, 12]. The second approach is to use web services technologies and use the standard messaging and standard interface definition mechanisms. The mentioned standard messaging and standard interface mechanisms are inherent to web services technologies and there is no need to predefine set of request/response in order to exposing geospatial resource over the web [10].

These two approaches are just standard approaches for exposing geospatial resources over the web. There are also other approaches [13] that utilize proprietary and platform-dependant solutions for exposing geospatial resources. This paper focuses on standard approaches. The first approach is standard and well supported in Geospatial Information (GI) community. The second approach belongs to the broader and more dynamic Information Technology (IT) community. As it mentioned before the second approach utilizes web services technologies that consist of several technologies such as XML, XSD, WSDL, SOAP as core technologies. These technologies can be used over the web (HTTP) or any other protocol. At the other hand the first approach (using geospatial services) limited to the web [10]. This is a serious issue in publishing geospatial resources to the users. Although some OGC specification can be defined using core web service technologies, but using core web service technologies for some OGC services is not possible (in standard manner) [14]. As an example Web

Map Service (WMS) specification is the most implemented geospatial service. It creates image of geospatial data (map in OGC terminology) as response to GetMap request over the HTTP. So the WMS service provides sharing of geospatial data at image-level. Since supported data types of web services defined by XSD and there is no native support for binary data in XSD, creating wrapper web service for WMS results in non-standard web service [10, 12, 13, 14 , 15]. The main geospatial service for sharing geospatial data at object level is Web Feature Service (WFS). WFS provides access to geospatial data using GML format which contains both geometrical as well as attribute properties of each geospatial data items [17]. Since GML is a XML grammar, there is no serious difference in using both kinds of services for publication and sharing of geospatial Big Data over the web at the object-level.

Often the huge volume of geospatial data is the reason for the complexity of publication of geospatial Big Data issue. In addition to huge volume of geospatial data, sometimes the velocity and variety components in geospatial Big Data are major reasons for the issue. For example in disaster management and when real-time or near-real time decision making is critical, it is necessary to access various voluminous geospatial data from different sources (satellite data, surveillance systems and social network data) with high frequency of change. In these situations the velocity of change of data and variety of data sources are as important as the volume of data.

Traditionally Relational Database Management Systems (RDBMS or SQL databases) with spatial extensions (Spatial Databases on top of relational or object-relational systems) were used as backend system for geospatial services [18]. Nowadays these systems still can be used in many geospatial data-related tasks but the mentioned systems are not efficient enough to handle geospatial Big Data, especially when the volume, velocity and variety of datasets are far beyond the capacity of a single server and the datasets need to be handled in distributed manner. As it illustrated in Figure 1, in a typical system (using the SQL database) there are four layers in the system for publication of geospatial data. In this case geospatial services (such as WFS) just provide access to geospatial data through a service layer for various kinds of clients. All the request and response processing is done in business logic layer. If geospatial datasets were stored in SQL or spatial databases in data layer, the business logic layer must contains a mapping layer. In the case of WFS and if the client requests to get data in GML format (application data model), the business logic layer must retrieve data from databases and then create a GML document.

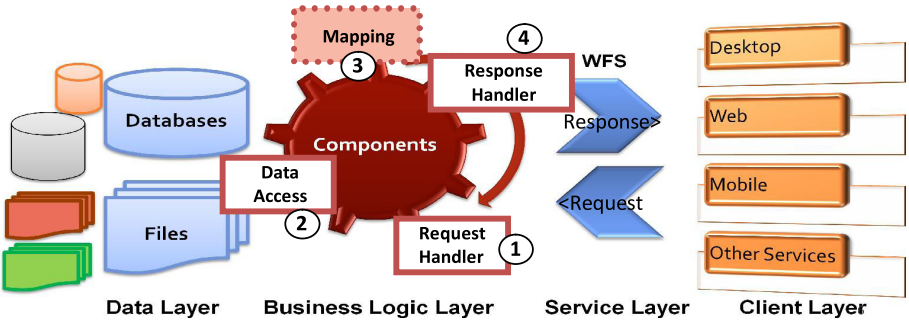

Fig. 1. A typical system for publication of geospatial data with relational or spatial databases

The mapping layer has negative effects on scalability, availability and performance of the system. In this case NoSQL databases can provide the required quality of services for standard publication of geospatial Big Data. In a nutshell, since the storage data model and application data model can be the same in NoSQL databases, the mapping layer is not necessary in the system. In addition, NoSQL databases are designed with the idea of distribution of data and processes in cluster of machines which is a major advantage in comparison with relational and object-relational databases [19, 20]. Following sections first explain the major types of NoSQL databases and then provide an evaluation of using a relational (SQL database), a spatial and a NoSQL for handling geospatial Big Data with focus on standard publication.

3 SQL and NoSQL Databases

Relational DBMSs (SQL databases) use tables, columns, keys and Structured Query Language (SQL) to perform all sorts of tasks with data. One of the important facets of SQL databases is the normalization process which ensure about storage of data items in separate tables and only once in whole database. The SQL databases usually are the best solutions when the schema of data is fixed and strong consistency is most needed feature. In other words, SQL databases are ideal solutions to managing structured data such that all users can access to the same set of data in same state at all times (strong consistency). The SQL databases can be effectively used in many common geospatial-related workflows. Since they support transaction and locking features, they provide efficient backend for enterprise GIS systems. Also geospatial data have a fixed schema and in most cases they are not used in isolation. As a result join of two or a few more datasets and connecting data through spatial operations is needed in most GIS workflows. For this reason managing fixed schema geospatial data with limited connectivity and using them in GIS workflows can usually be done effectively through SQL databases.

The SQL databases handle connected data using relationship and they retrieve connected data using joins. So connections between related data tables are stored using primary and foreign keys and join between connected tables are needed when retrieving data. So the related data are stored in SQL databases separately and they can be related using joins. However joins are one of the most computationally expensive processes for SQL databases. In most cases, joins are the bottleneck of SQL databases. In order to avoid many joins (which is needed in handling highly connected data in SQL databases) denormalization process can be used to store data items several times in single large tables. But there are several issues associated with denormalization process especially with providing consistency in large datasets. In addition to issues related to handling highly connected data, some other problems arise when SQL databases need to handle high volume of data and when scalability is needed by adding more servers and technologies to bind them together. When distribution is needed in SQL databases and with more loads on a SQL database, vertical partitioning and denormalization process are needed which results in complexity in providing strong consistency. In summary, to achieve high scalability in SQL databases the

normalized relational model of data storage has to be compromised and deviated from relational model.

The NoSQL (Not only SQL) DBMSs are a broad class of DBMSs identified by non-adherence to the SQL (relational) model. There are different types of NoSQL databases, each with distinct set of characteristics but they all can deal with large amounts of (semi-structured and unstructured) data and are able to support a large set of read and write operations and they are designed with scalability and distribution of data in mind. Since they are designed with distribution in mind, there are other alternatives for providing consistency (most notably eventual consistency model). For this reason NoSQL and relational models are not in contrast with each other rather they complement each other. The most widely accepted taxonomy of NoSQL databases are: key-value, document, column-family and graph [19].

The key-value database is the simplest type of NoSQL databases. As the name implies, this type of database stores schema-less data using keys. The key is usually a string and the stored values can be any valid type such as a primitive programming data type (string, integer etc) or a BLOB (Binary Large Object) without any predefined schema. It provides a simple API to access stored data. In most cases, this type of NoSQL database solution provides very little functionality beyond key-value storage. There is no support for relationships [20]. Queries are just limited to accessing values using keys but since there is one request to access the value, the queries are executed very quickly. Transactions are limited to a single key. The database contains no semantic model. Key-value databases can be utilized to store geospatial data but the complexity of geospatial data hinders spatial searches especially for polyline and polygon objects. For this reason, it needs to be spatially indexed for fast data retrieval which, in most cases, gives lower performance than a SQL or spatial database. In summary, key-value data stores are ideal for inserting, deleting and searching huge amount of simple data items using their unique identifiers (keys).

A document database in its simplest form is a key-value database in which the database understands its values [21]. In other words, values inside the database are based on predefined formats such as XML, JSON or BSON. This feature of document databases provides many advantages over key-value databases. Queries in this type of NoSQL databases are quite flexible. Similar to key-value databases, there is no need to adhere to a predefined schema to insert data. There is only limited support for relationships and joins as each document is stand alone. The document databases can be used for managing geospatial data more effectively than key-value databases. Since geospatial data inside the document database can be retrieved using flexible queries, they can be used for storing and managing geospatial data in multiple use cases. In fact many document databases support geospatial data natively or through extensions. Some of them can store geospatial data using GeoJSON format. Some queries such as proximity queries can be efficiently implemented using these document databases [22]. As mentioned before, relationships and joins are not supported the way they are supported in relational databases. The document-oriented nature of this NoSQL database has some major effects on the way that data can be retrieved. For example if the application needs data items from the same collection (documents with same schema) it would be very fast. However whenever the data items are part of different types of

documents there is no efficient approach to reduce the number of index-lookups. Special kinds of document database can partially handle relationships efficiently. XML document databases store documents as XML documents. This kind of document databases can be configured to enforce adherence to set of predefined XML schemas. In addition to all the advantages of document databases, XML document databases are able to utilize many XML technologies to provide further functionality. For example, they can use XQuery and XPath to perform various queries and create flexible result sets, they can make use of XPointer to reference other documents thus modeling a relationship and they can use XSD and RelaxNG to enforce schema validation [22]. Geography Markup Language (GML), as a standard mechanism for storing, modeling and exchanging geospatial data [17], is an XML-based grammar and so XML document databases are an ideal choice for managing geospatial data in GML format. Since the storage data model and application data model is XML document in XML document databases, they can be efficiently utilized when the standard publication of geospatial Big Data is needed.

The column-family databases store data in set of columns and distribute data based on columns. The column is the smallest unit of data and it is a triplet that contains a key, value and timestamp [19]. Column-family databases store all values beside the name of the columns and stores null values simply by ignoring the column. Usually, related columns compose a column-family. All the data in a single column family will be stored on the same physical set of files [20]. This feature provides higher performance for search, data retrieval and replication operations. A super column is a column that contains other columns but it cannot contain other super columns [21]. Most column-family databases use a distributed file-system to store data to disk and so provide a horizontally scalable system. In fact column-family databases are designed to run on a large number of machines. Queries in this type of NoSQL databases are limited to keys and in most cases they don't provide a way to query by column or value. By limiting queries to just keys, column-family databases ensure that procedure to find the machine containing actual data is quite fast. There is no join capability and, as in other types of NoSQL databases, there is limited support for transactions. Column-family databases are ideal for storing huge amounts of data when high availability is needed. Similar to document databases, there are many column-family databases which support the management and simple analysis of geospatial data. Any GIS related application which needs heavy data insertion and fast data retrieval with simple queries can efficiently make use of column-family databases. In summary this type of databases doesn't support relationships and in order to handle highly connected data, there is a need for mapping layer to create network structure (which is not efficient).

As the name implies graph databases are based on graph theory and employ nodes, properties and edges as their building blocks. The nodes and edges can have properties. In the graph databases various nodes might have different properties. The graph databases are well suited for data which can be modeled as networks such as road networks, social networks, biological networks and semantic webs. Their main feature is the fact that each node contains a direct pointer to its adjacent node, so no index lookups are necessary for traversing connected data. As a result they can manage huge amount of highly connected data since there is no need for expensive join

operations. Some of graph databases support transactions in the way that relational databases support them. In other words the graph database allows the update of a section of the graph in an isolated environment, hiding changes from other processes until the transaction is committed. Geospatial data can be modeled as graphs. Since graph databases support topology natively, topological relationship (especially connectivity) between geospatial data can be easily managed by this type of NoSQL databases [8]. In most GIS workflows, topological relationships play a major role. In addition since each edge in graph database can have different set of properties, they provide flexibility in traversal of network based on various properties. For example it is possible to combine time, distance, number of points of interest and user preferences in finding best path and the mentioned path would be unique for each user. In summary the storage model of graph databases is a graph and there is a need for mapping layer whenever other data structure is needed in application layer.

4 Implementation and Benchmarks

In order to find the best database model for standard publication of Big Data three systems were implemented based on the architecture illustrated in figure 1. Three different models of databases are used in the mentioned systems: relational (SQL), spatial and XML document. Since the evaluation is for database models rather than specific product, for consistency of the benchmark in this research Microsoft SQL Server 2012 (MS SQL Server) is used in three different models. The MS SQL Server is a relational DBMS (SQL database) with built-in support for spatial data using geography and geometry data types. In addition to spatial data types it has several advanced features that make MS SQL Server a capable spatial database (such as support for various spatial reference systems, diverse spatial indexing, implementation of OGC simple features specification and spatial topology handling based on 9 intersection model. Also the MS SQL Server has a native XML support through XML data type, XML indexing, support for XQuery and XPath and query optimization for XML queries. In other words, MS SQL Server is a spatial and XML document database on top of relational engine.

For the evaluation purpose, four geospatial datasets containing polygon features are created. The mentioned datasets contain hundred thousand (100k), one million (1m), ten million (10m) and a hundred million (100 m) polygon features. Each polygon feature has at least three points (three vertices) and at most 2000 vertices. In addition each polygon has at least one part and at most five parts (multi-part polygon). The EPSG 4326 was used as the spatial reference system to store coordinates of vertices.

Microsoft C# programming language was used for implementing business and service layer as well as client layer. The client application was a simple application for calling the WFS service to retrieve geospatial data based on several predefined queries. The mentioned application also utilized several profilers to record the metrics for the performance and scalability benchmarks.

4.1 Storage

Storage of polygon features (and multi-part polygon features) requires at least four tables in relational model. Figure 2, illustrates the conceptual model for the mentioned four tables using a ER diagram.

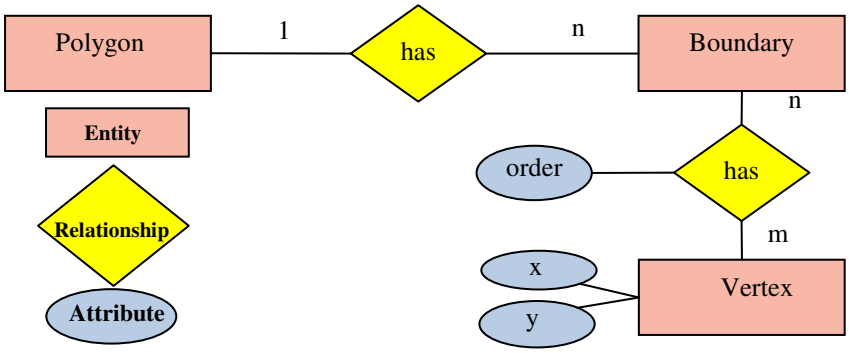

Fig. 2. Conceptual model of SQL Database for storing polygon features (in ERD)

A polygon in spatial database is defined using at least one closed ring. So there is one table for a polygon dataset. For XML document database, there is one GML document fragment per polygon (even multipart polygon), so there is one collection of documents for each polygon dataset.

4.2 Query Language and Retrieval

For relational and spatial databases the SQL language is utilized as the query language. As it mentioned before there are at least four tables for polygon features. As a result in order to retrieve data of a polygon three joins are needed. Since spatial database of this research implements the OGC Simple Features specification, all the spatial operators and methods are available as extension to standard SQL. There is no need for joins in spatial database since the polygon features are stored in a single table. The XML document database uses a XML-specific query language (XQuery) in order to retrieve geospatial data. In order to retrieve polygon features, there is no need for joins since the polygon features are stored as documents in a single collection.

4.3 Handling Attribute Relationships

Attribute relationships are non-spatial relationships (for example ownership relationship between an owner and a land). In relational and spatial databases this kind of relationship can be defined using primary and foreign keys. In order to retrieve related data in the mentioned databases, joins are needed. In XML document databases the attribute relationship can be defined by XPointer and for traversing between different document fragments XLink can be used. A link to one or more related documents can be found in the origin document fragment in XML document databases.

4.4 Topological Relationships

There is no support for evaluating topological relationships between geospatial data in SQL and XML document databases in MS SQL Server. Implementing methods for evaluating topological relationships or methods for simple analysis of geospatial data is similar in both XML document database and relational database. In order to implement the mentioned methods some in-memory data structures must be created. Since the data about single polygon are stored in just one collection, implementing such methods would be more efficient in XML document database than the relational database. In contrast, the built-in spatial extension of MS SQL Server implements all the methods for evaluating topological relationships and simple analysis outlined in OGC simple features.

4.5 Distribution

As data volume increase, it becomes more difficult and expensive to scale up (or vertical scalability; use a more expensive and bigger server to run the database on). A more efficient approach is scale out (or horizontal scalability; use of a cluster of servers to run a database). In general NoSQL databases are designed and implemented with focus on horizontal scalability and high volume of data. Depending on the distribution model, database can handle larger quantities of data and process a greater read and write traffic or more availability in the face of network slowdowns or breakages [19]. Usually there are two approaches for data distribution; replication and sharding. With replication copies of same data are stored on multiple nodes. So each bit of data can be found in multiple places [23]. In contrast, sharding puts different data on different nodes so each server acts as the single source for subset of data [20].

In the case of geospatial Big Data in most cases a combination of sharding and replication provides the highest availability, scalability and performance. Sharding in NoSQL database are easier since the natural unit of distribution is often the same as the unit of storage. In XML document database, distribution can be done based on the XML fragments. In other words, different XML fragments (storage unit) can be distributed on different nodes. In contrast sharding in SQL databases are not as straightforward as for the NoSQL database. The natural unit of distribution for geospatial Big Data is a geospatial feature. But data of a geospatial feature in SQL databases is spread over multiple tables which makes the distribution complex. For spatial database, the sharding is much easier than the SQL database but still is complicated in comparison with NoSQL database. It is possible to use replication for three models. But as it mentioned before, using different tables for single geospatial feature makes replication hard and complex for SQL database.

4.6 Schema Definition and Evolution

Schema definition for SQL and Spatial database is done using SQL language commands for creating tables, columns and indexes. For XML document databases the

XSD can be used for defining the schema of XML document. Both SQL and XSD support various data types. The schema change or evolution during the life cycle of application development is common and complex practice for databases. It is true that all NoSQL databases are schemaless (or schema-free) but in order to use the data inside NoSQL databases there is an implicit schema in the applications that use NoSQL databases. Table 1 summarizes the comparison between SQL, Spatial and NoSQL databases for handling geospatial Big Data.

Table 1. Various characteristics of three different approaches for geospatial Big Data storage

Item	SQL database	Spatial database	XML document NoSQL database
Logical Storage unit	Row	Geospatial feature	GML fragment
Logical Storage of a geospatial data-set	Multiple tables	Single table	Single collection
Query Language and Retrieval	SQL language	Extended SQL language with OGC Simple Features Specification	XQuery
Attribute Relationships	Primary and Foreign key and Joins	Primary and Foreign key and Joins	XLink and XPointer
Topological Relationships	No Native Support /Requires mapping layer	Extended SQL with OGC Simple Features Specification	No Native Support /Requires mapping layer
Distribution	Hard Replication Hard Sharding	Easier than SQL database Easier than SQL database	Easy Replication Easy Sharding
Schema Definition and Evolution	SQL language	SQL language	XSD language

4.7 Performance and Scalability Evaluations

In order to perform performance and scalability benchmarks, the mentioned three databases were filled with polygon datasets which include vast amount of features from 100,000 to 100,000,000 multi-part polygons. For performance tests, the response time was used as the metric. Figure 3 and 4, illustrate the results of the performance tests for single feature retrieval and feature retrieval using range queries respectively.

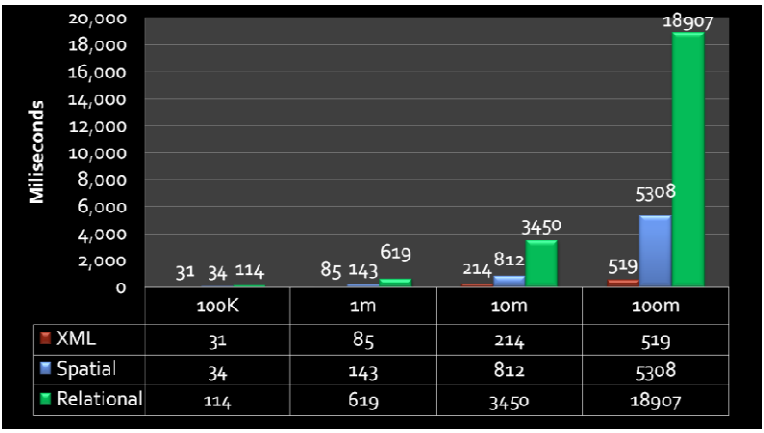

Fig. 3. Performance test for retrieval of single geospatial feature (lowest is the best)

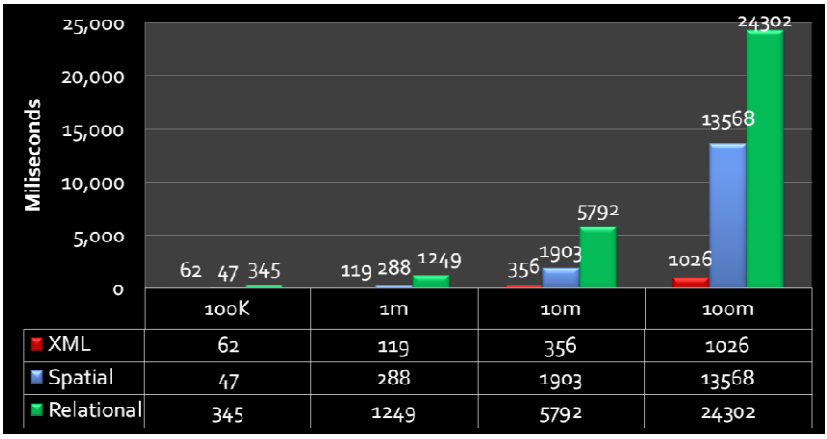

Fig. 4. Performance test for retrieval of group of geospatial features using various range queries (lowest is the best)

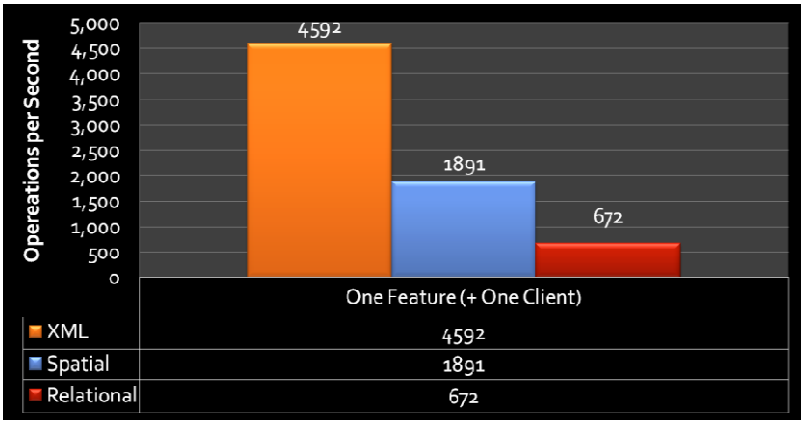

Fig. 5. Result of the scalability test (highest is the best)

In order to perform scalability benchmark, operation per second was utilized as metric. Figure 5 illustrates the result of the scalability test.

The results of tests proved that XML document database (NoSQL) provides better performance and scalability for standard publication of geospatial Big Data for some specific queries.

5 Conclusion

Geospatial data have specific characteristics that often reveal the limitation of computing systems. The storage and analysis of geospatial Big Data (high volume of high frequency of change geospatial data from various data sources) is challenging and complex and needs horizontal scalability and various models for consistency, data access and distribution. As the evaluations and benchmarks of this paper illustrate, the NoSQL databases provide several qualities needed for efficient analysis and management of geospatial Big Data. But this doesn't mean relational (SQL) or spatial databases don't have any place in geospatial Big Data landscape. The authors of this paper believe that polyglot geospatial data persistence approach is efficient model for geospatial Big Data handling. In other words using various database models for different tasks in a single system (polyglot data persistence).

References

1. Taniar, D., Rahayu, W.: A taxonomy for nearest neighbour queries in spatial databases. Journal of Computer and System Sciences 79(7), 1017–1039 (2013)
2. Amirian, P., Basiri, A., Alesheikh, A.: Interoperable exchange and share of urban services data through geospatial services and XML database. In: Complex, Intelligent and Software Intensive Systems, CISIS (2010)
3. Mahboubi, H., Bimonte, S., Deffuant, G., Chanet, J., Pinet, F.: Semi-Automatic Design of Spatial Data Cubes from Simulation Model Results. IJDWM 9(1), 70–95 (2013)
4. Basiri, A., Amirian, P., Winstanley, A.: The USE of Quick Response (QR) Code in Landmark-Based Pedestrian Navigation. International Journal of Navigation and Observation (2014)
5. Yildizli, C., Pedersen, T., Saygin, Y., Savas, E., Levi, A.: Distributed Privacy Preserving Clustering via Homomorphic Secret Sharing and Its Application to (Vertically) Partitioned Spatio-Temporal Data. IJDWM 7(1), 46–66 (2011)
6. Safar, M., Ebrahimi, D., Taniar, D.: Voronoi-based reverse nearest neighbor query processing on spatial networks. Multimedia Systems 15(5), 295–308 (2009)
7. Minelli, M., Chambers, M., Dhiraj, A.: Big Data, Big Analytics: Emerging Business Intelligence and Analytic Trends for Today's Businesses. Wiley (2013)
8. Amirian, P., Basiri, A., Winstanley, A.: Efficient Online Sharing of Geospatial Big Data Using NoSQL XML Databases. In Proceedings of IEEE Fourth International Conference on Computing for Geospatial Research and Application (COM. Geo) (2013)
9. Amirian, P., Basiri, A., Winstanley, A.: Implementing Geospatial Web Services Using Service Oriented Architecture and NoSQL Solutions. In: The Third International Conference on Digital Information and Communication Technology and its Applications (2013)

10. Amirian, P., Basiri, A., Alesheikh, A.: Standards-based, interoperable services for accessing urban services data for the city of Tehran. Computers, Environment and Urban Systems (2010)
11. Foerster, T., Schäffer, B.: A client for distributed geo-processing on the web. In: Ware, J.M., Taylor, G.E. (eds.) W2GIS 2007. LNCS, vol. 4857, pp. 252–263. Springer, Heidelberg (2007)
12. Sample, J., Shaw, K., Tu, S., Abdelguerfi, M.: Geospatial services and applications for the Internet. Springer (2008)
13. Stollberg, B., Zipf, A.: OGC Web Processing Service Interface for Web Service Orchestration Aggregating Geo-processing Services in a Bomb Threat Scenario. In: Ware, J.M., Taylor, G.E. (eds.) W2GIS 2007. LNCS, vol. 4857, pp. 239–251. Springer, Heidelberg (2007)
14. Schäffer, B., Baranski, B., Foerster, T., Brauner, J.: A Service-Oriented Framework for Real-time and Distributed Geoprocessing. In: Geospatial Free and Open Source Software in the 21st Century. Lecture Notes in Geoinformation and Cartography. Springer (2010)
15. Foerster, T., Schaeffer, B., Brauner, J., Baranski, B.: Geospatial Web Services for Distributed Processing - Applications and Scenarios. In: Geospatial Web Services: Advances in Information Interoperability, pp. 245–286. Information Science Reference (2011)
16. Vretanos, A.: OpenGIS Web Feature Service 2.0 Interface Standard, OGC 09-025r1 and ISO/DIS 19142 (2010)
17. Lake, R.: The application of geography markup language (GML) to the geological sciences. Computers & Geosciences 31(9), 1081–1094 (2005)
18. Oosterom, P.: Research and development in geo-information generalisation and multiple representation. Computers, Environment and Urban Systems (2010)
19. Fowler, M., Sadalage, P.: NoSQL Distilled. Addison Wesely (2013)
20. Celko, J.: Complete Guide to NoSQL, What Every SQL Professional Needs to Know about Non-Relational Databases. Morgan Kaufman (2014)
21. Chang, F., Dean, J., Ghemawat, S., Hsieh, W., Gruber, R.: Bigtable: A distributed storage system for structured data. In: Seventh Symposium on Operating System Design and Implementation (2006)
22. Amirian, P., Alesheikh, A.: Publishing Geospatial Data through geospatial web service and xml database system. American Journal of Applied Science 5(10) (2008)
23. Hecht, R., Jablonski, S.: NoSQL Evaluation A Use Case Oriented Survey. In: International Conference on Cloud and Service Computing, pp. 336–341 (2011)
24. McCreary, D., Kelly, A.: Making Sense of NoSQL. Manning (2013)